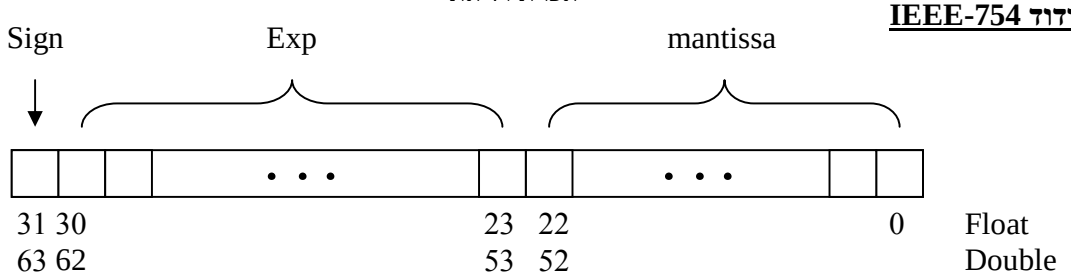
נוסחאון לקורס שיטות נומריות

אפרת וליאת

קידוד IEEE-754



המספר המיוצג ע"י שיטה זו- (בבסיס 2) $(-1)^s m*2^e$

כאשר-

s=sign

m=1.*מנטיסה*

e=Exp-127 (float) e=Exp-1023 (double)

הערה: כפל ב-2 של מספר בבסיס 2 שקול להזזת הנקודה מקום אחד ימינה.

ערכים מיוחדים שמורים ל- inf, -inf (infinity) ול-NaN (not a number)(לדוגמא 0/0). כאשר יש underflow, כלומר המספר קטן מדי בערך מוחלט לייצוג, אז *מנטיסה*.m=0.

כמו כן, קיים ייצוג כפול ל-0 והוא 0+,0-.

±inf : עבור float: s=±1 , m=0, e=128. עבור double: s=±1 , m=0, e=1024.

NaN: עבור float: $m \neq 0$,e=128. עבור double: $m \neq 0$,e=1024.

denormalized numbers: (*מנטיסה*.m=0): עבור float: Exp= 0, $m \neq 0$. e=Exp-126.

עבור double: Exp=0, $m \neq 0$. e=Exp-1022.

normalized numbers: (*מנטיסה*.m=1): עבור float: 127-<e<128. עבור double: 1023-<e<1024.

אפס: עבור float: e= -127, m=0. עבור double: e= -1023, m=0.

דיוק: עבור float הדיוק הוא בערך 2^{-23}, שזה בקירוב 10^{-7}. עבור double: 2^{-52}.

מגבלות:

הפעולות הרגילות אינן אסוציאטיביות, ואינן דיסטריביוטיביות.

המרה מבסיס 2 לבסיס 10:

ערכו של המספר בבסיס 2 המיוצג ע"י : $A_nA_{n-1}\ldots A_3A_2A_1.A_{-1}\ldots A_{-m}$ הוא: $\sum_{i=-m}^{n} 2^i \cdot Ai$ בבסיס 10.

המרה מבסיס 10 לבסיס 2:

מפרקים את המספר לשלם ולשבר.

המרת השלם:

מחלקים את המספר ב-2 וכותבים את השארית. את השלם שנשאר מחלקים שוב ב-2 וכותבים את השארית מתחת לשארית הקודמת. כך ממשיך התהליך עד אשר נשאר שלם שהוא אפס. המספר בבסיס 2 הוא השאריות שנקראות מלמטה למעלה.

דוגמא:

נחשב את 23 בבסיס 2:

	שלם	שארית
23:2	12	1
12:2	6	0
6:2	3	0
3:2	1	1
1:2	0	1

$(23)_{10} = (11001)_2$

המרת השבר:

כופלים את המספר ב-2 וכותבים את השלם. את השבר שנשאר כופלים שוב ב-2 וכותבים את השלם מתחת לשלם הקודם. כך ממשיך התהליך עד אשר נשאר שבר שהוא אפס. המספר בבסיס 2 הוא השלמים שנקראים מלמעלה למטה.

דוגמא:

נחשב את 0.375 בבסיס 2:

	שבר	שלם
0.375*2	0.75	0
0.75*2	0.5	1
0.5*2	0	1

$(0.375)_{10} = (0.011)_2$

שגיאות:

שגיאות מדידה:

שגיאה מוחלטת:

$$||\Delta x|| = ||x - \bar{x}||$$

שגיאה יחסית:

$$\delta x = \frac{||\Delta x||}{||x||}$$

נורמות של וקטורים עבור וקטורי עמודה:

$$||\vec{x}||_1 = \sum_{i=1}^{n} |x_i| \qquad ||\vec{x}||_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \qquad ||\vec{x}||_\infty = \max(x_i)$$

עבור שגיאה מוחלטת:

$$\Delta(x+y) = \left|\frac{\partial(x+y)}{\partial x}\right| \cdot \Delta x + \left|\frac{\partial(x+y)}{\partial y}\right| \cdot \Delta y = |1| \cdot \Delta x + |1| \cdot \Delta y = \Delta x + \Delta y$$

$$\Delta(x-y) = \left|\frac{\partial(x-y)}{\partial x}\right| \cdot \Delta x + \left|\frac{\partial(x-y)}{\partial y}\right| \cdot \Delta y = |1| \cdot \Delta x + |-1| \cdot \Delta y = \Delta x + \Delta y$$

$$\Delta(x \cdot y) = \left|\frac{\partial(x \cdot y)}{\partial x}\right| \cdot \Delta x + \left|\frac{\partial(x \cdot y)}{\partial y}\right| \cdot \Delta y = |y| \cdot \Delta x + |x| \cdot \Delta y$$

$$\Delta\left(\frac{x}{y}\right) = \left|\frac{\partial\left(\frac{x}{y}\right)}{\partial x}\right| \cdot \Delta x + \left|\frac{\partial\left(\frac{x}{y}\right)}{\partial y}\right| \cdot \Delta y = \left|\frac{1}{y}\right| \cdot \Delta x + \left|-\frac{x}{y^2}\right| \cdot \Delta y$$

עבור שגיאה יחסית:

$$\delta(x+y) = \left|\frac{\Delta(x+y)}{x+y}\right| = \frac{\Delta x + \Delta y}{|x+y|} = \frac{|x|}{|x+y|}\frac{\Delta x}{|x|} + \frac{|y|}{|x+y|}\frac{\Delta y}{|y|} = \frac{|x|}{|x+y|}\delta x + \frac{|y|}{|x+y|}\delta y$$

$$\delta(x-y) = \left|\frac{\Delta(x-y)}{x-y}\right| = \frac{\Delta x + \Delta y}{|x-y|} = \frac{|x|}{|x-y|}\frac{\Delta x}{|x|} + \frac{|y|}{|x-y|}\frac{\Delta y}{|y|} = \frac{|x|}{|x-y|}\delta x + \frac{|y|}{|x-y|}\delta y$$

$$\delta(x \cdot y) = \delta x + \delta y$$

$$\delta(x / y) = \delta x + \delta y$$

טעות עיגול:

$$\frac{||\Delta x||}{||x||} \approx O\left(\varepsilon_{machine}\right) \rightarrow \text{float } 2^{-23}, \quad \text{double } 2^{-52}$$

$$||\Delta f(x)|| \approx \varepsilon_{machine} \cdot ||f(x)||$$

טעות קיצוץ (קירוב, אלגוריתם):

כתוצאה מפיתוח לטור טיילור:

$$R_{n+1} = \frac{f^{(n+1)}(c) \cdot (x-a)^{n+1}}{(n+1)!}$$

$$c \epsilon [a, x]$$

אפרת וליאת

<u>טעות מתפשטת (פונקציה):</u>

בפונקציה חד מימדית: $\Delta f(x) = |f'(x)| \cdot \Delta x$

בפונקציה רב מימדית: $||\Delta f(x)|| = \sum_{i=1}^{n} \left| \frac{\partial}{\partial x_i} f \right| \cdot \Delta x_i$

במספר פונקציות רב מימדיות: $||\Delta f(x)|| \le ||\vec{J}|| \cdot ||\overrightarrow{\Delta x}||$

$$\frac{||\overrightarrow{\Delta f(x)}||}{||\overrightarrow{f(x)}||} \le \frac{||\vec{J}|| \cdot ||\overrightarrow{\Delta x}||}{||\overrightarrow{f(x)}||}$$

$$\vec{J} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1 & \frac{\partial}{\partial x_2} f_1 & \cdot & \cdot & \cdot & \frac{\partial}{\partial x_n} f_1 \\ \frac{\partial}{\partial x_1} f_2 & \cdot & & & & \frac{\partial}{\partial x_n} f_2 \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot \\ \frac{\partial}{\partial x_1} f_n & \frac{\partial}{\partial x_2} f_n & \cdot & \cdot & \cdot & \frac{\partial}{\partial x_n} f_n \end{bmatrix}$$

פקודה ב-matlab: J= jacobian(f,v)

וקטור המשתנים

נורמות של מטריצות:

$$||A_{mxn}||_1 = \max_j \left(\sum_{i=1}^{m} |a_{ij}| \right)$$

$$||A_{mxn}||_\infty = \max_i \left(\sum_{j=1}^{n} |a_{ij}| \right)$$

$$||A_{mxn}||_2 = \sqrt{\max(\lambda)}$$

כאשר λ ערך עצמי של $A^* A$

פקודה ב-matlab: Norm(A, *נורמה*)

הנורמה היא 1 2 או inf

טעות כוללת: סכום שלוש הטעויות האחרונות.

אפרת וליאת

שיטות למציאת שורשים:

שיטת החצייה:

שיטה זו מבוססת על משפט ערך הביניים האומר שאם פונקציה רציפה בקטע סגור [a,b] אזי לכל γ המקיימת $f(a) \le \gamma \le f(b)$ (נניח בה"כ $f(a) \le f(b)$) קיים $c \in [a, b]$ עבורו f(c) = γ, ובפרט אם γ = 0 אזי אם $f(a)\cdot f(b) < 0$ אז קיים $c \in [a, b]$ עבורו f(c) = 0.
ניתן להקטין בכל פעם את ההסתכלות לקטע המוכל בקטע המקורי כל עוד הוא מקיים את הדרישה ש- $f(a')\cdot f(b') < 0$.
את זאת נעשה ע"י חלוקה של הקטע לשניים, אם חלקו הראשון לא מקיים את התנאי סימן שבוודאות קיים שורש בחלק השני. גם את הקטע החדש נחלק לשניים, ונבחר את החלק שמקיים את התנאי. ניתן להשתמש בשיטה זו על מנת לקבל ניחוש התחלתי קרוב לשורש ואז להשתמש בשיטה מהירה יותר.
מלבד העובדה שהתכנסות שיטה זו, גם אם וודאית, היא איטית בדרך כלל, לא ניתן לזהות בה שורשים כפולים (המופיעים מספר זוגי של פעמים) כמו בפונקציה:

$$f(x) = (x-3)\cdot(x-1)^2$$

מכיוון שהפונקציה משיקה לציר ה-x בשורש 1, ולא חותכת אותו, כלומר לא ניתן לבחור a,b שיתחמו את השורש כך שיתקיימו התנאים הדרושים.

אלגוריתם:

1. נגדיר $c = \frac{a+b}{2}$
   - א. אם f(c) = 0 הפסק.
   - ב. אם $f(a)\cdot f(c) < 0$ b = c.
   - ג. אחרת a = c.

   חזור ל-1

קריטריונים לעצירה:

1. $|f(a)-f(c)| < \varepsilon_{machine}$ או $|f(b)-f(c)| < \varepsilon_{machine}$
2. $|f(c)| < \varepsilon_{machine}$
3. מספר הצעדים שנעשו גדול מ-N.

מימוש האלגוריתם ב-matlab:

```
function root = bisection1(f,a,b,n)
c=(a+b)/2;
num=1;
while (abs(f(c))>eps && num<n)
    if f(a)*f(c)<0
        b=c;
    else
        a=c;
    end
    c=(a+b)/2;
    num=num+1;
end
root = c;
end
```

$\varepsilon_{machine} = 2^{-52} = 2.2204\cdot 10^{-16}$

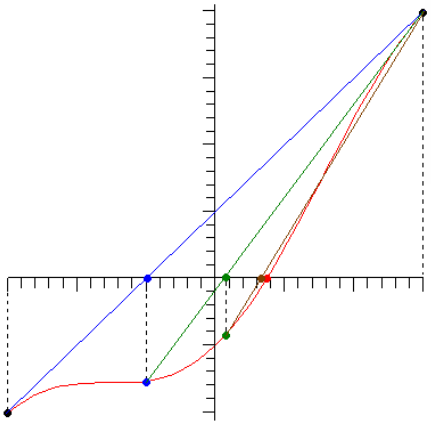
שיטת רגולה פלסי:

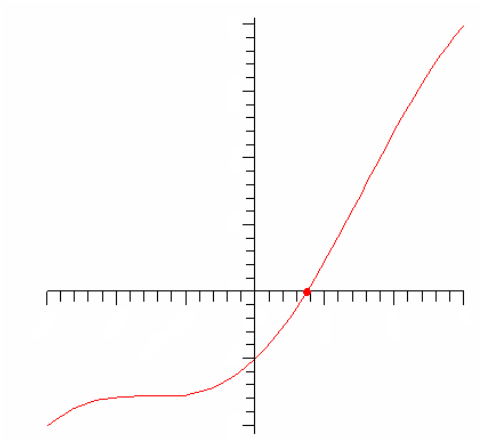
בדומה לשיטת החצייה נבחר קטע [a,b], כך ש $f(a)\cdot f(b) < 0$, ונקבל שהוא תוחם את השורש.
בשיטה זו ניצור מיקום מדומה של השורש ע"י העברת ישר בין ערכי הפונקציה בקצות הקטע ונבחר את הנקודה בה הישר חותך את ציר ה-x, נסמנה c. נבחר את אחד משני הקטעים שהתקבלו, [a,c] או [b,c] אם $f(a)\cdot f(c) < 0$ או אם $f(b)\cdot f(c) < 0$, בהתאמה.
נעביר ישר נוסף בין שני ערכי הפונקציה בקצות הקטע ונבחר את הנקודה בה הישר חותך את ציר ה-x.

אלגוריתם:

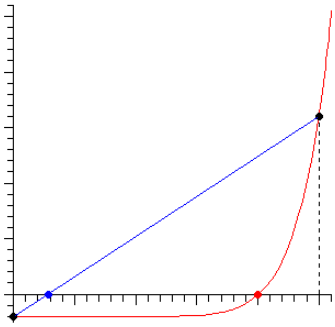
1. נגדיר $c = \dfrac{a\cdot f(b) - b\cdot f(a)}{f(b) - f(a)}$
   - א. אם f(c) = 0 הפסק.
   - ב. אם $f(a)\cdot f(c) < 0$ b = c.
   - ג. אחרת a = c.
   - חזור ל-1.

דוגמא:





בדרך כלל, שיטה זו תתכנס מהר יותר משיטת החצייה.
בשיטה זו אנו בוחרים את נקודת הקטע הבא בצורה "חכמה יותר", נקבל שמיקום השורש המדומה יהיה קרוב יותר לקצה הקטע שערך הפונקציה בו קרוב יותר לאפס, ולא בדיוק במרכזו, כמו שמתקיים בשיטת החצייה. תוצאה זו לא תמיד מדויקת, כמו לדוגמא בפונקציה הבאה:



$x^{10} - 1$

ניתן לראות שיידרש מספר רב של צעדים על מנת להתקרב לשורש, ובמקרה זה שיטת החצייה אף תהיה מהירה יותר.
קיבלנו שהמיקום המדומה שהתקבל קרוב יותר לאפס, כשלמעשה, השורש קרוב יותר לנקודה השנייה.

מימוש האלגוריתם ב-matlab:

```
function root=regulafalsi(f,a,b,n)
c=(a*f(b)-b*f(a))/(f(b)-f(a));
num=1;
while(abs(f(c))>eps && num<n)
    if (f(a)*f(c)<0)
        b=c;
    else
        a=c;
    end
    c=(a*f(b)-b*f(a))/(f(b)-f(a));
    num=num+1;
end
root=c;
end
```

אפרת וליאת

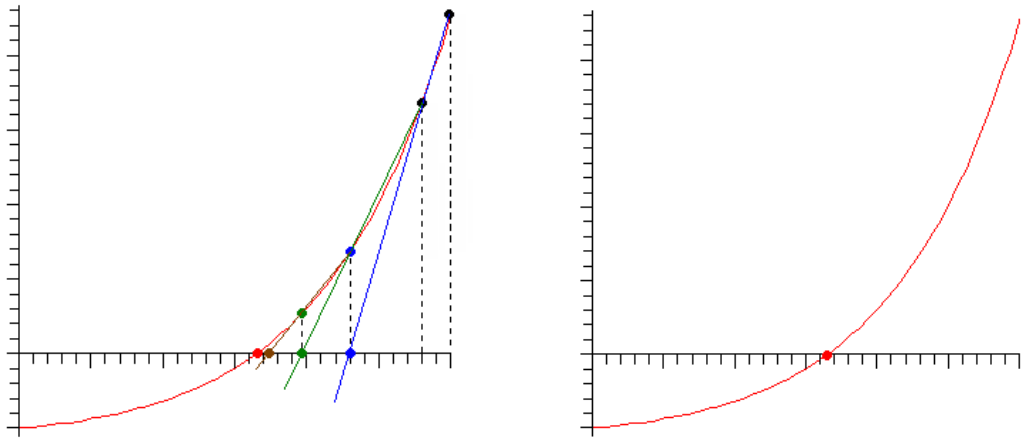
שיטת המיתר:

בדומה לשיטת רגולה פלסי גם כאן אנו מעבירים בכל פעם קו ישר בין ערכי הפונקציה של שתי נקודות ובחירת הנקודה החדשה להיות זו שבה הישר חותך את ציר ה-x. נקודות ההתחלה נבחרות ללא תנאים, ואף יכולות להיות שתיהן בצד אחד של השורש, אולם סדר בחירתן חשוב.

$$x_{n+1} = x_n - \frac{f(x_n)\cdot(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} = \frac{x_{n-1}\cdot f(x_n) - x_n\cdot f(x_{n-1})}{f(x_n) - f(x_{n-1})}$$

נוסחא זו אמורה להתכנס אל השורש.
ההבדל בין שתי השיטות הוא שכאן מגדירים נוסחת נסיגה וכל נקודה חדשה מחליפה את קודמתה לפי הסדר, ללא הצבת תנאים.
בניגוד לשיטת החצייה ורגולה פלסי, בשיטה זו לא תוחמים את השורש, והיא לא בהכרח תתכנס. כאשר היא מתכנסת, היא בדרך כלל מתכנסת מהר יותר מרגולה פלסי. סדר התכנסות 1.6

דוגמא:



מימוש השיטה ב-matlab:

```
function root=secant(f,x0,x1,n)
x2=x1-f(x1)*(x1-x0)/(f(x1)-f(x0));
num=1;
while (abs(f(x2))>eps && num<n)
    x0=x1;
    x1=x2;
    x2=x1-f(x1)*(x1-x0)/(f(x1)-f(x0));
    num=num+1;
end
root =x2;
end
```

גם בשיטה זו קיימת בעיה דומה במקרה של שורשים כפולים, באופן דומה נגדיר פונקציה חדשה $u(x) = \frac{f(x)}{f'(x)}$ ונפעיל את שיטת המיתר על u במקום על f.

$$x_{n+1} = x_n - \frac{u(x_n)\cdot(x_n - x_{n-1})}{u(x_n) - u(x_{n-1})}$$

שיטת ניוטון-רפסון:

בשיטה זו בוחרים נקודה התחלתית ומעבירים משיק לפונקציה בנקודה זו. נקודת החיתוך בין המשיק לציר ה-x אמורה להוות קירוב טוב יותר לשורש.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

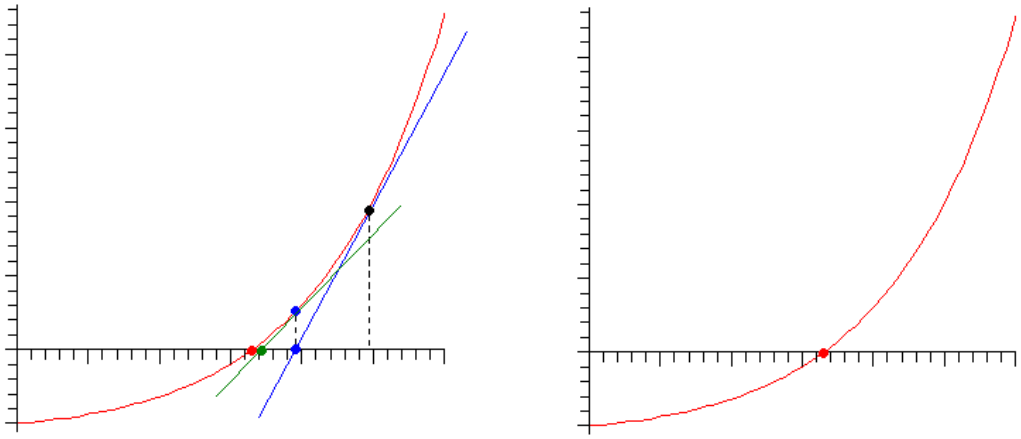
שיטה זו לא בהכרח מתכנסת, למשל כאשר הנגזרת באחת מסדרת הנקודות שואפת לאפס. בדומה לשיטת המיתר, גם כאן לא תוחמים את השורש. סדר ההתכנסות הוא 2.

שיטה זו מתכנסת בדרך כלל מהר יותר משאר השיטות. אך כאמור, ישנן פונקציות שעבורן ההתכנסות איטית יותר, כדוגמת הפונקציה: $x^{10}-1$

השגיאה:

$$e_{n+1} \simeq \left| \frac{f''(\alpha)}{2f'(\alpha)} \right| \cdot e_n^{\,2}$$

דוגמא:



מימוש השיטה ב-matlab:

```
function root=nr(f,x0,n)
syms x;
y=inline(diff(f(x)));
x=x0-f(x0)/y(x0);
num=1;
while (abs(f(x))>eps && num<n)
    x0=x;
    x=x0-f(x0)/y(x0);
    num=num+1;
end
root=x;
end
```

פונקציות נוספות שעבורו ההתכנסות היא איטית הן פונקציות בעלות שורש כפול בנקודה מסוימת, לדוגמא $f(x) = (x-3)\cdot(x-1)^2$ ועבור השורש 1, ההתכנסות תהיה לינארית. במצבים כאלו ניתן לשנות את השיטה ע"י הגדרת פונקציה חדשה $u(x) = \frac{f(x)}{f'(x)}$ ואז להפעיל את ניוטון רפסון על u במקום על f. לאחר פישוט, נקבל:

$$x_{n+1} = x_n - \frac{f(x_n)\cdot f'(x_n)}{(f'(x_n))^2 - f(x_n)\cdot f''(x_n)}$$

שיטה זו מתכנסת עם סדר 2, והיא יעילה יותר עבור שורשים כפולים. במקרה של שורשים רגילים היא פחות יעילה מהשיטה המקורית, בנוסף לעובדה שהיא מצריכה יותר חישובים.

שיטת ניוטון-רפסון רב מימדית:

תנאי הכרחי: הפיכות היעקוביאן.

$$\overrightarrow{x_{n+1}} = \overrightarrow{x_n} - J(x_n)^{-1} \cdot \overrightarrow{f(\overrightarrow{x_n})}$$

שיטת נקודת שבת:

בשיטה זו נחפש משוואה כדי לנבא מיקום משוער של השורש, ניתן לפתח נוסחא כזאת ע"י סידור מחדש של המשוואה $f(x)=0$ לקבל $x=\varphi(x)$.

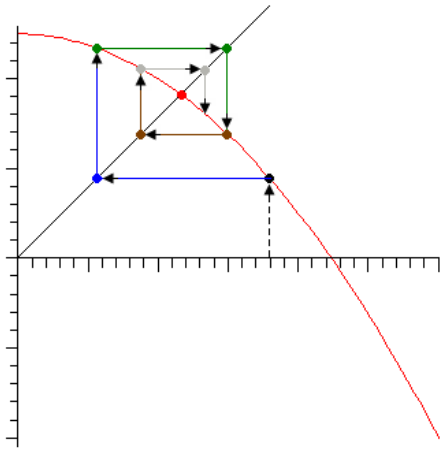
שיטת נקודת שבת תתכנס אם

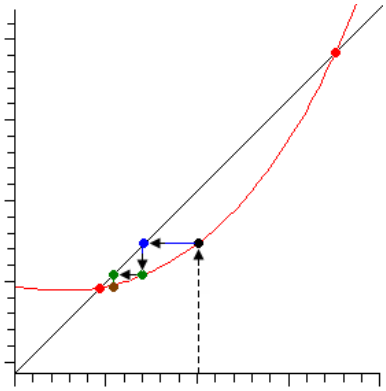
$$|\varphi'(\alpha)| < 1 \qquad \text{כאשר} \qquad f(\alpha)=0$$

החישוב יעשה בצורה הבאה:

$$x_{n+1}=\varphi(x_n)$$

דוגמא:





קצב התכנסות:

$$\lim_{n\to\infty}\frac{e_{n+1}}{e_n^{\,p}}=q=\left|\frac{\varphi^{(p)}(\alpha)}{p!}\right|$$

כאשר יש התכנסות, סדר ההתכנסות p, הוא המספר הטבעי הראשון, עבורו:

$$\varphi^{(p)}(\alpha)\neq 0\ ,\ \varphi^{(1)}(\alpha)=\cdots=\varphi^{(p-1)}(\alpha)=0$$

מימוש השיטה ב-matlab:

```
function root= conmap(phi,x0,n)
x=phi(x0);
num=1;
while (abs(phi(x)-x)>eps && num<n)
    x=phi(x);
    num=num+1;
end
root = x;
end
```

שיטת נקודת שבת רב מימדית:

השיטה תתכנס כאשר: $||J(\vec{x})|| < 1$

$$\begin{bmatrix} x_{1_{n+1}} \\ x_{2_{n+1}} \\ \vdots \\ x_{k_{n+1}} \end{bmatrix} = \begin{bmatrix} \varphi_1(x_{1_n},\cdots x_{k_n}) \\ \varphi_2(x_{1_n},\cdots x_{k_n}) \\ \vdots \\ \varphi_k(x_{1_n},\cdots x_{k_n}) \end{bmatrix} \qquad \overrightarrow{x_{n+1}}=\varphi(\overrightarrow{x_n})$$

אלגברה לינארית בשיטות נומריות:

$$A\vec{x} = \vec{b}$$

ההשפעה של טעות ב-x על b:

$$\frac{1}{||A||\cdot||A^{-1}||}\cdot\frac{||\Delta x||}{||x||} \le \frac{||\Delta b||}{||b||} \le \underbrace{||A||\cdot||A^{-1}||}_{\text{Condition number}}\cdot\frac{||\Delta x||}{||x||}$$

הפקודה ב-matlab: c = cond(X,*נורמה*)

השפעת טעות ב-b על x:

$$\frac{1}{||A||\cdot||A^{-1}||}\cdot\frac{||\Delta b||}{||b||} \le \frac{||\Delta x||}{||x||} \le ||A||\cdot||A^{-1}||\cdot\frac{||\Delta b||}{||b||}$$

השפעת טעות ב-A על x:

$$\frac{||\Delta x||}{||x||} \le ||A||\cdot||A^{-1}||\cdot\frac{||\Delta A||}{||A||}$$

Pivoting:

החלפת שורות, כך שהאיבר המוביל יהיה הגדול ביותר (בערך מוחלט), ובעזרתו נאפס את שאר העמודה שמתחתיו. פעולה זו מצמצמת השפעה של טעויות עיגול, ומשמשת כפיתרון חלקי למטריצות עם condition גבוה (ill condition).

שיטת גאוס:

בשיטה זו מדרגים את המטריצה עד שמתקבלת מטריצה משולשית עליונה ואז פותרים בעזרת הצבה לאחור, כלומר מוצאים את x_n, מציבים אותו במשוואה שלפניו ומוצאים את x_{n-1}, את שניהם מציבים במשוואה שקדמה לזאת ומוצאים את x_{n-2} וכך ממשיכים עד שמוצאים את כל הערכים של וקטור הפתרונות.

בשיטה זו נדרשות פחות פעולות מאשר בדירוג מלא, אך היא לא טובה ממנה בהרבה.

מימוש השיטה עם pivoting ב-matlab:

```
function x=gaussel (A,b)
n=size(A,1);
for j=1:(n-1)
    maxind=j;
    for m=(j+1):n
        if(abs(A(m,j))>A(maxind,j))
            maxind=m;
        end
    end
    temp=A(maxind,:);
    A(maxind,:)=A(j,:);
    A(j,:)=temp;
    temp2=b(maxind);
    b(maxind)=b(j);
    b(j)=temp2;
    for i =(j+1):n
        if(A(j,j)~=0)
            b(i)=b(i)-A(i,j)*b(j)/A(j,j);
            A(i,:)=A(i,:)-A(i,j)*A(j,:)/A(j,j);
        end
    end
end
for i= n:-1:1
    sum=0;
    for j=n:-1:(i+1)
        sum=sum+x(j)*A(i,j);
    end
    x(i)=(b(i)-sum)/A(i,i);
end
end
```

Pivoting

דירוג (שילוש)

הצבה לאחור

הפקודה ב-matlab:

מחזיר פתרון למשוואה Ax=b : x=A\b

פירוק LU:

בפירוק זה A = LU כאשר A מטריצה ריבועית.

משולשית עליונה — משולשית תחתונה

U היא המטריצה המתקבלת מדירוג A למטריצה משולשית עליונה. $U = e_{\frac{n\cdot(n-1)}{2}} \cdot \ldots \cdot e_2 \cdot e_1 \cdot A$

L היא המטריצה ההופכית למכפלת המטריצות האלמנטריות. $L = e_1^{-1} \cdot e_2^{-1} \cdot \ldots \cdot e_{\frac{n\cdot(n-1)}{2}}^{-1}$

Ax=b

LUx=b

Ly=b כאשר Ux=y. נמצא את y לפי הצבה קדימה, ואז נמצא את x לפי הצבה לאחור.

דוגמא:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad R_2 - \boxed{\frac{a_{21}}{a_{11}}} \cdot R_1 \to R_2$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} - \frac{a_{21}}{a_{11}} \cdot a_{12} & a_{23} - \frac{a_{21}}{a_{11}} \cdot a_{13} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad R_3 - \boxed{\frac{a_{31}}{a_{11}}} R_1 \to R_3$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} - \frac{a_{21}}{a_{11}} \cdot a_{12} & a_{23} - \frac{a_{21}}{a_{11}} \cdot a_{13} \\ 0 & a_{32} - \frac{a_{31}}{a_{11}} \cdot a_{12} & a_{33} - \frac{a_{31}}{a_{11}} \cdot a_{13} \end{bmatrix} \quad R_3 - \boxed{\frac{a_{32} - \frac{a_{31}}{a_{11}} \cdot a_{12}}{a_{22} - \frac{a_{21}}{a_{11}} \cdot a_{12}}} \cdot R_2 \to R_3$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \boxed{\frac{a_{21}}{a_{11}}} & 1 & 0 \\ \boxed{\frac{a_{31}}{a_{11}}} & \boxed{\frac{a_{32} - \frac{a_{31}}{a_{11}} \cdot a_{12}}{a_{22} - \frac{a_{21}}{a_{11}} \cdot a_{12}}} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} - \frac{a_{21}}{a_{11}} \cdot a_{12} & a_{23} - \frac{a_{21}}{a_{11}} \cdot a_{13} \\ 0 & 0 & a_{33} - \frac{a_{31}}{a_{11}} \cdot a_{13} - \frac{a_{32} - \frac{a_{31}}{a_{11}} \cdot a_{12}}{a_{22} - \frac{a_{21}}{a_{11}} \cdot a_{12}} \cdot \left(a_{23} - \frac{a_{21}}{a_{11}} \cdot a_{13} \right) \end{bmatrix}$$

אפרת וליאת

פירוק LU עם pivoting:

PA=LU כאשר P מטריצת חילופי השורות, $A=P^tLU$. נפתור PAx=Pb, באותה דרך כמו פירוק ללא pivoting, רק עבור Pb.

הפקודה ב-matlab: [L U P] = lu(A)

מימוש הפירוק עם pivoting ב-matlab:

```
function [P,L,U]= ludecom(A)
n=size(A,1);
P=eye(n);
L=eye(n);
for j=1:(n-1)
   maxind=j;
    for m=(j+1):n
        if(abs(A(m,j))>A(maxind,j))
            maxind=m;
        end
    end
    temp=A(maxind,:);
    A(maxind,:)=A(j,:);
    A(j,:)=temp;
    temp=P(maxind,:);
    P(maxind,:)=P(j,:);
    P(j,:)=temp;
    temp=L(maxind,1:(j-1));
    L(maxind,1:(j-1))=L(j,1:(j-1));
    L(j,1:(j-1))=temp;
    for i =(j+1):n
        if(A(j,j)~=0)
            L(i,j)=A(i,j)/A(j,j);
            A(i,:)=A(i,:)-A(i,j)*A(j,:)/A(j,j);
        end
    end
end
U=A;
end
```

<u>פירוק כולסקי:</u>

עבור מטריצה A סימטרית ($A=A^t$) וחיובית ממש ניתן לפרק את המטריצה באופן הבא: $A=RR^t$

מטריצה משולשית תחתונה

מטריצה חיובית ממש אמ"מ מתקיים אחד מהתנאים הבאים:

א. $x^tAx>0$ לכל וקטור שורה x בגודל n השונה מאפס.

ב. כל הערכים העצמיים של A גדולים ממש מאפס.

ג. כל המינורים הראשיים של A גדולים מאפס.

המשך-

לפי פירוק LU $A = L\cdot U$, שקול ל-$A = L\cdot \mathrm{D}\cdot U'$ כאשר:

$$U = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix} \qquad U' = \begin{bmatrix} 1 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ 0 & 1 & \frac{a_{23}}{a_{22}} & \vdots \\ \vdots & \ddots & \ddots & \frac{a_{n-1\,n}}{a_{n-1\,n-1}} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \qquad \mathrm{D} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}$$

ניתן להוכיח כי $L^t = U'$ כלומר $A=LDL^t$

הערכים על האלכסון של D חיוביים ולכן ניתן לרשום $A = \underbrace{L\cdot\sqrt{\mathrm{D}}}_{\mathrm{R}}\cdot\underbrace{\sqrt{\mathrm{D}}\cdot L^t}_{\mathrm{R}^t}$

הפקודה ב-matlab: B=chol(A)

מחזירה את המטריצה R^t

<u>פירוק QR:</u>

$$A = Q\cdot R$$

מטריצה אורתוגונלית — משולשית עליונה

תהי מטריצה A, $A = \begin{pmatrix} | & | & & | \\ \vec{a_1} & \vec{a_2} & \cdots & \vec{a_n} \\ | & | & & | \end{pmatrix}$ נגדיר: $c = ||a_1||_2$ $\quad u = \begin{bmatrix} a_{11}-c \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} \quad \vec{v} = \frac{u}{||u||}$

$$||u|| = \sqrt{(a_{11}-c)^2 + a_{21}{}^2 + \cdots + a_{n1}{}^2} = \sqrt{(a_{11}-c)^2 + c^2 - a_{11}{}^2} =$$
$$\sqrt{a_{11}{}^2 - 2a_{11}\cdot c + c^2 + c^2 - a_{11}{}^2} = \sqrt{2c^2 - 2a_{11}\cdot c}$$

נגדיר $H_1 = I - 2V\cdot V^t$. V וקטור עמודה מנורמל.

נוכיח H_1 מטריצה סימטרית: $H_1{}^t = (I-2V\cdot V^t)^t = I^t - 2(V^t)^t\cdot V^t = I - 2V\cdot V^t = H_1$

נוכיח H_1 מטריצה אורתוגונלית:

$$H_1\cdot H_1{}^t = (I-2V\cdot V^t)\cdot(I-2V\cdot V^t) = I - 4V\cdot V^t + 4V\cdot V^t\cdot V\cdot V^t = I - 4V\cdot V^t + 4V\cdot V^t = I$$

נוכיח: $H_1\cdot\vec{a_1} = \alpha\cdot e_1$ (α כלשהי).

$$H_1\cdot\vec{a_1} = (I-2V\cdot V^t)\cdot\vec{a_1} = I\cdot\vec{a_1} - 2V\cdot V^t\cdot\vec{a_1} = \vec{a_1} - 2V\cdot V^t\cdot\vec{a_1} =$$

המשך-

$$= \vec{a_1} - \frac{2}{\sqrt{2c^2 - 2a_{11}\cdot c}} \cdot \vec{v} \cdot \frac{1}{\sqrt{2c^2 - 2a_{11}\cdot c}} \cdot \begin{bmatrix} a_{11} - c & a_{21} & \cdots & a_{n1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} =$$

$$= \vec{a_1} - \frac{2}{2c^2 - 2a_{11}\cdot c} \cdot \vec{v} \cdot \left((a_{11} - c)\cdot a_{11} + a_{21}{}^2 + \cdots + a_{n1}{}^2 \right) =$$

$$= \vec{a_1} - \frac{1}{c^2 - a_{11}\cdot c} \cdot \vec{v} \cdot \left(c^2 - c\cdot a_{11} \right) = \vec{a_1} - \vec{v} = \begin{bmatrix} c \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

כנדרש.

קיבלנו:

$$H_1 \cdot A = \begin{bmatrix} c & \\ 0 & \boxed{\quad * \quad} \\ \vdots & \\ 0 & \end{bmatrix}$$

באופן דומה, נפעיל את התהליך על תת מטריצה הכוכביות, ונמצא את H_2 ... H_{n-1}, כאשר נרחיב את המטריצות הנ"ל למטריצות מסדר nxn, ע"י הוספת המספר הדרוש של שורות ועמודות בהתאמה, באופן הבא:

$$\begin{array}{cc} & \overbrace{\hphantom{1 \quad 0 \quad \cdots \quad \cdots \quad \cdots \quad 0}}^{n} \\ n \left\{ \right. & \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & & & & \vdots \\ \vdots & & 1 & 0 & \cdots & 0 \\ \vdots & & 0 & & & \\ \vdots & & \vdots & & \boxed{H_i} & \\ 0 & \cdots & 0 & & & \end{bmatrix} \end{array}$$

ניתן להוכיח באינדוקציה שמטריצה R המוגדרת ע"י $H_{n-1} \cdot H_{n-2} \cdot \ldots \cdot H_1 \cdot A = R$ היא מטריצה משולשית עליונה. בנוסף, מכפלת H_1 *...*H_{n-1} היא אורתוגונלית, ולכן:

$$A = \left(H_{n-1} \cdot H_{n-2} \cdot \ldots \cdot H_1 \right)^t \cdot R = \underbrace{H_1 \cdot H_2 \cdot \ldots \cdot H_{n-1}}_{Q} \cdot R$$

הפקודה ב-matlab: [Q R]=qr(A)

מימוש הפירוק ב-matlab:

```
function [Q,R]=qrdecom(A)
n=size(A,1);
Q=eye(n);
H(:,:,:)=zeros(n,n,n-1);
v=zeros(n,1);
R=A;
for i=1:(n-1)
    H(:,:,i)=eye(n);
    c=norm(R(i:n,i));
    u=R(:,i);
    u(i)=u(i)-c;
    v(i:n)=u(i:n)/norm(u(i:n));
    H(i:n,i:n,i)=eye(n-i+1)-2*v(i:n)*(v(i:n))';
    Q=Q*H(:,:,i);
    R=H(:,:,i)*R;
end
end
```

מערך של n-1 מטריצות בגודל nxn
$H_{n-1},\dots,H_1$

"הרחבנו" מראש את כל אחת מ-H_i המטריצות,
ולאחר מכן נשנה רק תת מטריצה של כל אחת מהן.

הסתכלות על תת "מטריצת הכוכביות"

מציאת וקטור הפתרונות למשוואה $A\vec{x}=\vec{b}$ באמצעות פירוק QR :

Ax=b
QRx=b
$Rx=Q^t b$

ומכיוון ש-R מטריצה משולשית עליונה נפתור בהצבה לאחור.

<u>פירוק SVD:</u>

בפירוק זה A מיוצגת ע"י: $A_{mxn}=U_{mxm}\cdot S_{mxn}\cdot V_{nxn}{}^{t}$

כאשר U היא מטריצה של וקטורים עצמיים של AA^*
S היא מטריצה של ערכים עצמיים משותפים של A^*A ושל AA^*
V היא מטריצה של וקטורים עצמיים של A^*A

A^*A מטריצה מסדר nxn
AA^* מטריצה מסדר mxm

(הערה: מטריצה סימטרית תמיד ניתנת ללכסון משמע יש לה n וקטורים עצמיים שונים)

U ו-V הן מטריצות אורתוגונליות.

דוגמא עבור A מסדר 2x4:

$$\underbrace{\begin{array}{c}2\\ \\ \end{array}\left\{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\end{bmatrix}\right.}_{4} = \underbrace{\begin{bmatrix} | & | \\ u_1 & u_2 \\ | & | \end{bmatrix}}_{2} \cdot \underbrace{\begin{bmatrix} \lambda_1 & 0 & 0 & 0\\ 0 & \lambda_2 & 0 & 0\end{bmatrix}}_{4} \cdot \underbrace{\begin{bmatrix} \text{——} & v_1{}^t & \text{——}\\ \text{——} & v_2{}^t & \text{——}\\ \text{——} & v_3{}^t & \text{——}\\ \text{——} & v_4{}^t & \text{——}\end{bmatrix}\left.\begin{array}{c}4\\ \\ \\ \\ \end{array}\right\}}_{4}$$

הפקודה ב-matlab: [U S V] = svd(A)

אפרת וליאת

המשך-

מציאת וקטור הפתרונות למשוואה $A\vec{x} = \vec{b}$ באמצעות SVD:

$$Ax = b$$
$$USV^t x = b$$
$$SV^t x = U^t b$$
$$V^t x = S^{-1} U^t b$$
$$x = VS^{-1} U^t b$$

מטריצה U היא אורתוגונלית.
החישוב של מטריצה הופכית למטריצה אלכסונית הוא פשוט.
מטריצה V היא אורתוגונלית.

<u>שיטות איטרטיביות למציאת וקטור פתרונות:</u>

$$A_{nxm} \cdot \vec{x}_{nx1} = \vec{b}_{nx1}$$
$$(L + D + U) \cdot \vec{x} = \vec{b}$$

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{n1} & \cdots & a_{n\,n-1} & 0 \end{bmatrix} \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & a_{nn} \end{bmatrix} \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \ddots & a_{n-1\,n} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

<u>שיטת יעקובי:</u>

$$(L + D + U)\vec{x} = b$$
$$Dx + (L + U)\vec{x} = b$$
$$\vec{x} + D^{-1} \cdot (L + U)\vec{x} = D^{-1} \cdot b$$
$$\vec{x} = -D^{-1} \cdot (L + U)\vec{x} = D^{-1} \cdot b$$

$$\boxed{\overrightarrow{x_{n+1}} = -D^{-1}(L + U)\overrightarrow{x_n} + D^{-1} \cdot b}$$

$$H = -D^{-1} \cdot (L + U)$$

<u>שיטת גאוס-זיידל:</u>

$$(L + D + U)\vec{x} = b$$
$$(L + D)\vec{x} + U\vec{x} = b$$
$$\vec{x} + (L + D)^{-1} U\vec{x} = (L + D)^{-1} \cdot b$$
$$\vec{x} = -(L + D)^{-1} U\vec{x} + (L + D)^{-1} \cdot b$$

$$\boxed{\overrightarrow{x_{n+1}} = -(L + D)^{-1} U\overrightarrow{x_n} + (L + D)^{-1} \cdot b}$$

$$H = -(L + D)^{-1} U$$

<u>תנאים להתכנסות שתי השיטות:</u>

הע"ע המקסימלי של H בערכו המוחלט קטן מ-1.

או

למטריצה A קיים אלכסון דומיננטי- $\forall\, i \quad |a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad \wedge \quad |a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ji}|$

אפרת וליאת

:Power Method

זוהי שיטה איטרטיבית למציאת הע"ע המקסימלי של מטריצה.
נבחר x_0 התחלתי (בדרך כלל $[1\ 1\ ..\ 1]^t$), ונחשב בכל פעם את הוקטור החדש שיתקבל ע"י $x_{n+1} = A \cdot x_n$
את הוקטור הזה ננרמל, על התהליך הזה נחזור עד שההפרש בין הוקטור החדש לוקטור הקודם יהיה מספיק קטן.
התוצאה שתתקבל תשאף לו"ע המתאים לע"ע המקסימלי, הנורמה של וקטור זה לפני הנרמול תשאף לע"ע המקסימלי.

מימוש השיטה ב-matlab:

```
function [eig,v]=powem(A,x0,p,n)
x1=A*x0;
eig = norm(x1,p);
x1=x1/eig;
num=1;
while (norm(x1-x0,p)>eps &&num<n)
    x0=x1;
    x1=A*x0;
    eig = norm(x1,p);
    x1=x1/eig;
    num=num+1;
end
v=x1;
```

הנורמה שבה נרצה לנרמל

אם הריבוי האלגברי של הע"ע המקסימלי גדול מאחד אז השיטה תהיה פחות יעילה.

אינטרפולציה:

עבור פולינום:
כאשר נתונות n+1 נקודות $[x_0, y_0],\ldots,[x_n, y_n]$ אזי קיים פולינום יחיד מדרגה לא גדולה מ-n, כך ש- $P(x_i) = y_i$.

שיטה ראשונה:

נקבע $\Phi_0, \Phi_1,\ldots, \Phi_n$ כך ש- $\Phi_k(x) = x^k$
יש למצוא $\alpha_0, \alpha_1,\ldots, \alpha_n$ כך שיתקיים:

$$\forall\, i \qquad f(x_i) = \sum_{k=0}^{n} \alpha_i \cdot \Phi_k(x_i) = y_i$$

בהצגה כמטריצה:

$$\begin{array}{c} \begin{array}{ccccc} \Phi_0 & \Phi_1 & & & \Phi_n \end{array} \\ \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ 1 & x_3 & x_3^2 & \cdots & x_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \end{array} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}$$

ומחשבים את וקטור הפתרונות.
הבעיה בשיטה זו היא שה-condition number של מטריצת וונדרמונדה הוא גבוה, ולכן טעות קטנה במטריצה תתבטא בטעות גדולה בפיתרון.

אפרת וליאת

שיטה שנייה (שיטת לגרנז'):

נגדיר $\Phi_j(x) = L_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$

ניתן לראות שמתקיים:

$$L_j(x_i) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

$$\begin{bmatrix} \Phi_0(x_0) & \Phi_1(x_0) & \cdots & \Phi_n(x_0) \\ \Phi_0(x_1) & \Phi_1(x_1) & & \vdots \\ \vdots & & \ddots & \vdots \\ \Phi_0(x_n) & \cdots & \cdots & \Phi_n(x_n) \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

n+1 נקודות, פולינום ממעלה n.
n+1 פונקציות בסיס.

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$\Downarrow$$

$$\alpha_k = y_k$$

לכן פולינום האינטרפולציה $P(x) = \sum_{j=0}^{n} y_j \cdot L_j(x)$.

יתרונות: שינוי ערך ה-y באחת מנקודות הדגימה או יותר, ניתן להיעשות בקלות יחסית.
חסרונות: הוספה או החסרה של נקודת דגימה אחת או יותר, מצריכה עדכון של כל פונקציות הבסיס.

שיטה שלישית (שיטת ניוטון):

נגדיר $\Phi_i(x) = \prod_{j=0}^{i-1} (x - x_j)$

המטריצה המתקבלת כאן היא זו:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & \Phi_1(x_1) & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ \vdots & \vdots & & \ddots & 0 \\ 1 & \Phi_1(x_n) & \cdots & \cdots & \Phi_n(x_n) \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

המשך-
הפרשים מחולקים:

$$
\begin{array}{lll}
x_0 \quad y_0 & & \\
 & \dfrac{y_1-y_0}{x_1-x_0}=y_{10} & \\
x_1 \quad y_1 & & \dfrac{y_{21}-y_{10}}{x_2-x_0}=y_{210} \\
 & \dfrac{y_2-y_1}{x_2-x_1}=y_{21} & \\
x_2 \quad y_2 & & \dfrac{y_{321}-y_{210}}{x_3-x_0}=y_{3210} \\
 & \dfrac{y_3-y_2}{x_3-x_2}=y_{32} & \\
x_3 \quad y_3 & & \dfrac{y_{32}-y_{21}}{x_3-x_1}=y_{321}
\end{array}
$$

כאשר בכל פעם $y_{i(i-1)\ldots(k+1)k}=\dfrac{y_{i\ldots(k+1)}-y_{i-1\ldots k}}{x_i-x_k}$.

הפולינום שיתקבל הוא:

$$P(x)=y_0+y_{10}\overbrace{(x-x_0)}^{\Phi_1}+y_{210}\overbrace{(x-x_0)(x-x_1)}^{\Phi_2}+\ldots+y_{n(n-1)\ldots 210}\overbrace{(x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_{n-1})}^{\Phi_n}$$

(במקרה זה המקדמים הם המסלול העליון, אך ניתן לבחור כל מסלול שהוא).

יתרונות: הוספה או החסרה של נקודת דגימה חדשה מצריכה פחות חישובים.
חסרונות: החישוב מסובך יותר מחישוב הפולינום לפי לגרנז'.

<u>פולינום הרמיט:</u>

נתונות n+1-k נקודות, k נגזרות. מעלת הפולינום שיתקבל תהיה לא גדולה מ-n. חישוב הפולינום יעשה בדומה לחישוב לפי שיטת ההפרשים המחולקים.

$$
\begin{array}{lll}
x_0 \quad y_0 & & \\
 & f'(x_0) & \\
x_0 \quad y_0 & & \dfrac{\dfrac{y_1-y_0}{x_1-x_0}-f'(x_0)}{x_1-x_0} \\
 & \dfrac{y_1-y_0}{x_1-x_0} & \\
x_1 \quad y_1 & & \ldots \\
 & f'(x_1) & \\
x_1 \quad y_1 & &
\end{array}
$$

את $\Phi_0, \Phi_1,\ldots, \Phi_n$ נחשב באופן דומה לפי המסלול הנבחר, כאשר לדוגמא, אם נבחר את המסלול העליון נקבל

$$\Phi_1=(x-x_0)$$

$$\Phi_2=(x-x_0)\cdot(x-x_0)$$

$$\vdots$$

הפקודה ב-matlab: מתאימה פולינום, לפי המעלה המבוקשת, לסדרת הנקודות:
P=polyfit ([x_0,..., xn], [y_0,…, yn], <u>*מעלה*</u>)

:Cubic spline

בשיטה זו של אינטרפולציה, נחשב פולינום $S_i(x)$ ממעלה שלישית בין כל שתי נקודות $[x_i,y_i],[x_{i+1},y_{i+1}]$. נדרוש רציפות הפונקציה ורציפות הנגזרות הראשונה והשנייה. על מנת לקבל פתרון יחיד יש לדרוש שיתקיימו עוד שני תנאים. כאשר האינטרפולציה היא Natural cubic spline נדרוש שהנגזרות השניות בנקודות x_0 ו-x_n יהיו אפס.
הנוסחא לפולינום היא:

$$S_i(x)=\frac{a_i\cdot(x_{i+1}-x)^3}{6h_i}+\frac{a_{i+1}\cdot(x-x_i)^3}{6h_i}+\left(\frac{y_i}{h_i}-\frac{a_i\cdot h_i}{6}\right)\cdot(x_{i+1}-x)+\left(\frac{y_{i+1}}{h_i}-\frac{a_{i+1}\cdot h_i}{6}\right)\cdot(x-x_i)$$

כאשר נחשב $a_0..a_n$ ע"י מערכת המשוואות הבאה:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{6}\cdot h_0 & \frac{1}{3}\cdot(h_0+h_1) & \frac{1}{6}\cdot h_1 & 0 & 0 \\ 0 & \frac{1}{6}\cdot h_1 & \frac{1}{3}\cdot(h_1+h_2) & \frac{1}{6}\cdot h_1 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}=\begin{bmatrix} 0 \\ \frac{(y_2-y_1)}{h_1}-\frac{(y_1-y_0)}{h_0} \\ \frac{(y_3-y_2)}{h_2}-\frac{(y_2-y_1)}{h_1} \\ \vdots \\ 0 \end{bmatrix}$$

יתרונות:
בביצוע אינטרפולציה עבור מספר רב של נקודות הפולינום שיתקבל יהיה בדרך כלל ממעלה גבוהה, לעומת אינטרפולציה מסוג cubic spline בה הפולינומים מוגבלים למעלה שלישית.
כאשר מעלת הפולינום גבוהה מתקבלות תנודות חדות, בניגוד לאינטרפולציה מסוג cubic spline שם התנודות מינימליות.
באמצעות אינטרפולציה מסוג spline ניתן לקרב גם עקומות שאינן פונקציות.

חסרונות:
לא ניתן לבצע אקסטרפולציה, כלומר הערכת התנהגות הפונקציה מחוץ לטווח המדידות.

הפקודה ב-matlab עבור cubic : yi = interp1(x,Y,xi, 'spline')
כאשר x סדרת נקודות, אשר מקבלות ערכי Y, yi היא פונקצית האינטרפולציה בנקודות xi

קירובים:

שיטת הריבועים המזעריים:

כאשר קיימת שגיאה בערכים שקיבלנו עבור נקודות המדידה, אינטרפולציה לא תמיד תיתן קירוב מהימן להתנהגות האמיתית של הפונקציה, ולכן נעדיף למצוא קירוב לפונקציה אשר לא בהכרח יעבור דרך כל הנקודות, אלא יספק את הקירוב המתאים ביותר.

נתונות $[x_1, y_1],\ldots,[x_n, y_n]$ נרצה להעביר פולינום כך ש- $P(x_i)\approx y_i$.

עבור P(x)=ax+b:

$$\begin{bmatrix} \sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i \\ \sum_{i=1}^{n}x_i & \sum_{i=1}^{n}1 \end{bmatrix}\cdot\begin{bmatrix} a \\ b \end{bmatrix}=\begin{bmatrix} \sum_{i=1}^{n}y_i\cdot x_i \\ \sum_{i=1}^{n}y_i \end{bmatrix}$$

אפרת וליאת

המשך-

עבור $P(x)=ax^2+bx+c$:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \cdot x_i^2 \\ \sum_{i=1}^{n} y_i \cdot x_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}$$

ה-condition number של מטריצה זו הוא גבוה, ולכן נשתמש בפולינומים אורתוגונליים.

<u>פולינומים אורתוגונליים:</u>

בשיטה זו ניצור קירוב לנקודות ע"י פולינום ממעלה m:

$$P(x) = \sum_{k=1}^{m+1} \alpha_k \cdot \Phi_k(x) = \alpha_1 \cdot \Phi_1(x) + \alpha_2 \cdot \Phi_2(x) + \ldots + \alpha_{m+1} \cdot \Phi_{m+1}(x)$$

<u>קירוב לסדרת נקודות:</u>

כאשר:

- $\sum_{k=1}^{n} \Phi_i(x_k) \cdot \Phi_j(x_k) = 0$ לכל $i \neq j$
- כל Φ_i הוא פולינום מתוקן.
- כל Φ_i הוא ממעלה i-1.

על מנת לחשב את המקדמים, נפתור את מערכת המשוואות הבאה:

$$\begin{bmatrix} \sum_{i=1}^{n} \left(\Phi_1(x_i)\right)^2 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sum_{i=1}^{n} \left(\Phi_{m+1}(x_i)\right)^2 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \vdots \\ \vdots \\ \alpha_{m+1} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \cdot \Phi_1(x_i) \\ \vdots \\ \vdots \\ \sum_{i=1}^{n} y_i \cdot \Phi_{m+1}(x_i) \end{bmatrix}$$

ונקבל:

$$\alpha_k = \frac{\sum_{i=1}^{n} y_i \cdot \Phi_k(x_i)}{\sum_{i=1}^{n} \left(\Phi_k(x_i)\right)^2}$$

אפרת וליאת

קירוב לפונקציות:

$$\int_a^b \Phi_i(x)\cdot\Phi_j(x)\mathrm{d}x = 0 \quad \text{לכל} \quad i \neq j$$

כאשר פונקצית המשקל w(x)=1:

$$\alpha_k = \frac{\int_a^b f(x)\cdot\Phi_k(x)\mathrm{d}x}{\int_a^b \left(\Phi_k(x)\right)^2\mathrm{d}x}$$

עבור פונקצית משקל כללית w(x) :

$$\alpha_k = \frac{\int_a^b f(x)\cdot\Phi_k(x)\cdot w(x)\mathrm{d}x}{\int_a^b \left(\Phi_k(x)\right)^2\cdot w(x)\mathrm{d}x}$$

פולינומי לז'נדר:

סדרה של פולינומים אורתוגונליים לפי לז'נדר:

- עבור פונקציה בקטע [-1,1].
- פונקצית משקל w(x)=1.
- נוסחא רקורסיבית:

$$P_0 = 1 \qquad P_1 = x$$
$$(n+1)\cdot P_{n+1}(x) = (2n+1)\cdot x\cdot P_n(x) - n\cdot P_{n-1}(x)$$

- נוסחא לא רקורסיבית:

$$P_n(x) = \frac{1}{2^n\cdot n!}\cdot\frac{\mathrm{d}^n}{\mathrm{d}\,x^n}\left(x^2-1\right)^n$$

- חישוב המקדמים:

$$\int_{-1}^{1}\left(P_n(x)\right)^2\cdot 1\cdot\mathrm{d}x = \frac{2}{2n+1}$$

$$\alpha_n = \frac{2n+1}{2}\cdot\int_{-1}^{1} f(x)\cdot P_n(x)\,\mathrm{d}x$$

- כאשר מחשבים קירוב לפונקציה בקטע [a,b] נשתמש בנוסחת המרה: $y = \frac{2x-a-b}{b-a}$
נחשב קירוב עבור y, ולבסוף נציב חזרה.

אפרת וליאת

המשך-

פולינומי לז'נדר (עד מעלה שמינית):

$$
\begin{aligned}
P_0 &= 1 \\
P_1 &= x \\
P_2 &= \frac{3}{2}x^2 - \frac{1}{2} \\
P_3 &= \frac{5}{2}x^3 - \frac{3}{2}x \\
P_4 &= \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8} \\
P_5 &= \frac{63}{8}x^5 - \frac{35}{4}x^3 + \frac{15}{8}x \\
P_6 &= \frac{231}{16}x^6 - \frac{315}{16}x^4 + \frac{105}{16}x^2 - \frac{5}{16} \\
P_7 &= \frac{429}{16}x^7 - \frac{693}{16}x^5 + \frac{315}{16}x^3 - \frac{35}{16}x \\
P_8 &= \frac{6435}{128}x^8 - \frac{3003}{32}x^6 + \frac{3465}{64}x^4 - \frac{315}{32}x^2 + \frac{35}{128}
\end{aligned}
$$

כאשר n זוגי, $p_n(x)$ פונקציה זוגית, ולהפך.

<u>פולינומי צ'בישב:</u>

סדרה של פולינומים אורתוגונליים לפי צ'בישב:

- עבור פונקציה בקטע [-1,1].
- פונקצית משקל

$$w(x) = \frac{1}{\sqrt{1-x^2}}$$

- נוסחא רקורסיבית:

$$T_0 = 1 \qquad T_1 = x$$
$$T_{n+1} = 2x \cdot T_n - T_{n-1}$$

- נוסחא לא רקורסיבית:

$$T_n = \cos(n \cdot \arccos(x))$$

- חישוב המקדמים:

$$\int_{-1}^{1} \frac{\left(T_n(x)\right)^2}{\sqrt{1-x^2}} \cdot \mathrm{d}x = \begin{cases} \pi & n = 0 \\ \frac{\pi}{2} & n > 0 \end{cases}$$

$$\alpha_n = \begin{cases} \frac{1}{\pi} \cdot \int_{-1}^{1} \frac{f(x) \cdot T_n(x)}{\sqrt{1-x^2}} \mathrm{d}x & n = 0 \\ \frac{2}{\pi} \cdot \int_{-1}^{1} \frac{f(x) \cdot T_n(x)}{\sqrt{1-x^2}} \mathrm{d}x & n > 0 \end{cases}$$

המשך-

- כאשר מחשבים קירוב לפונקציה בקטע [a,b] נשתמש בנוסחת המרה: $y=\frac{2x-a-b}{b-a}$
נחשב קירוב עבור y, ולבסוף נציב חזרה.

פולינומי צ'בישב (עד מעלה שמינית):

$$
\begin{aligned}
T_0 &= 1\\
T_1 &= x\\
T_2 &= 2x^2-1\\
T_3 &= 4x^3-3x\\
T_4 &= 8x^4-8x^2+1\\
T_5 &= 16x^5-20x^3+5x\\
T_6 &= 32x^6-48x^4+18x^2-1\\
T_7 &= 64x^7-112x^5+56x^3-7x\\
T_8 &= 128x^8-256x^6+160x^4-32x^2+1
\end{aligned}
$$

כאשר n זוגי, $T_n(x)$ פונקציה זוגית, ולהפך.

<u>דיפרנציאציה נומרית:</u>

נגזרת קדמית: $f'(x)=\frac{f(x+\Delta x)-f(x)}{\Delta x}$

נגזרת אחורית: $f'(x)=\frac{f(x)-f(x-\Delta x)}{\Delta x}$

נגזרת מרכזית: $f'(x)=\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}$

<u>אינטגרציה נומרית:</u>

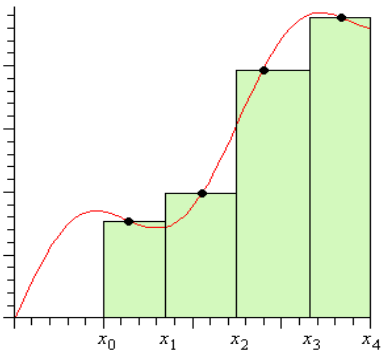
<u>סכומי רימן-</u>

נבחר n נקודות ונחשב את $\sum_{k=1}^{n} f(y_k)\cdot\Delta x_k$

כאשר y_k נקודה אקראית בקטע $[x_{k-1},x_k]$

$$\Delta x_k = x_k - x_{k-1}$$

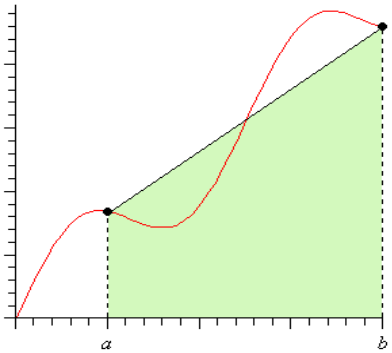
ככל שנקטין את Δx_k נקבל דיוק יותר גבוה, אך זה מצריך מספר רב יותר של נקודות.



אפרת וליאת

כלל הטרפז:

עבור צעד אחד:
בשיטה זו נעביר בין ערכי הפונקציה בקצות הקטע קו ישר (למעשה נבצע אינטרפולציה עבור 2 נקודות), נבצע אינטגרציה על קו זה ונקבל למעשה את שטח הטרפז כמתואר בשרטוט:



$$\int_a^b f(x)\,dx \approx \frac{h}{2}[f(a)+f(b)]$$

כאשר $h=b-a$

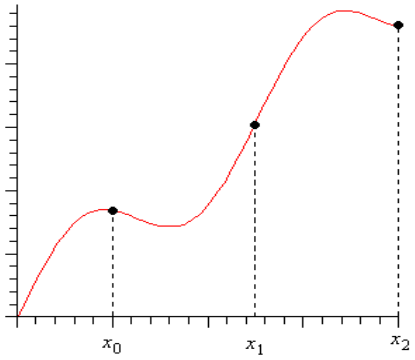
על מנת לקבל דיוק טוב יותר נחלק את הקטע ל-n קטעים שווים, ובכל קטע נחשב את שטח הטרפז שיתקבל.
הנוסחא לכלל טרפז עם n צעדים:

$$T_n=\frac{h}{2}\left[f(a)+2\sum_{i=1}^{n-1}f(a+ih)+f(b)\right]$$

כאשר $h=\frac{a+b}{n}$

שגיאה: $-\frac{(b-a)\cdot f''(c)}{12}\cdot h^2 \qquad a\le c\le b$

שיטת סימפסון:



בשיטה זו נבצע אינטרפולציה עבור 3 נקודות: קצות הקטע ומרכזו.
יתקבל פולינום שמעלתו לא עולה על 2.
נבצע אינטגרציה על קירוב זה ונקבל את כלל סימפסון:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}[f(x_0)+4f(x_1)+f(x_2)]$$

כאשר $h=\frac{b-a}{2}$

על מנת לקבל דיוק טוב יותר נחלק כל קטע לשני קטעים (הוספת נקודה במרכז הקטע) בכל צעד ונבצע עבור כל 3 נקודות בקטע קירוב לפי נוסחא זו.
נקבל:

$$\int_a^b f(x)dx = \frac{h}{3}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+2f(x_4)+\ldots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)]$$

$h=\frac{b-a}{n}$

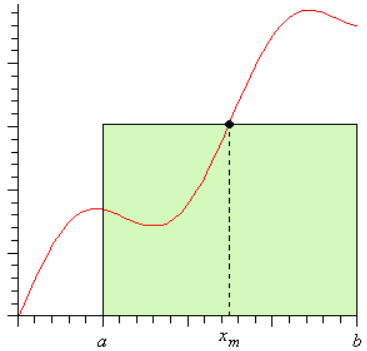
שגיאה: $-\frac{(b-a)f^{(4)}(c)}{180}\cdot h^4 \qquad a\le c\le b$

הפקודה ב-matlab: q = quad(fun,a,b)

אפרת וליאת

Midpoint Rule:

בשיטה זו נחשב את שטחו של מלבן שרוחבו כרוחב הקטע וגובהו שווה לערך הפונקציה במרכז הקטע.



$$\int_a^b f(x)\,\mathrm{d}x \approx (b-a)\cdot f(x_m)$$

כאשר $x_m = \frac{b+a}{2}$

על מנת לקבל דיוק טוב יותר נחלק את הקטע ל-n קטעים שווים וסוכמים את שטחי המלבנים אשר מתקבלים מהכפלת אורך הקטע h בערך הפונקציה במרכז הקטע.

$$M_n = \sum_{i=0}^{n-1} f\left(\frac{x_i + x_{i+1}}{2}\right)\cdot h$$

שגיאה חסומה ע"י $O(h^2)$.

שיטת גאוס:

בשיטה זו נדגום את הפונקציה ב-n+1 נקודות, ונבצע אינטרפולציה לפי לגרנז'. לאחר מכן נוכל לבצע אינטגרציה על הפולינום שיתקבל, אשר מעלתו לא עולה על n.
בשיטה זו ניתן לחשב מראש את פולינומי לגרנז' עבור סדרת נקודות דגימה מסוימת, לבצע אינטגרציה על כל אחד מהם בנפרד, ולאחר מכן לכפול בערכי ה-y השונים בהתאמה. כך ניתן לקבל נוסחה כללית עבור כל פונקציה הנדגמת באותן הנקודות, כאשר משתנים ערכי ה-y בלבד.
מעשית, כאשר מספר הנקודות רב, ומעלת הפולינום המתקבל גבוהה, מתקבלות תנודות חדות בפולינום, ולכן שיטה זו אינה שימושית.

תרבועי גאוס:

בשיטה זו נדגום את הפונקציה בשורשי פולינום לז'נדר מסדר n, על מנת לחשב את האינטגרל של הפונקציה בקטע [-1,1]. נבצע אינטרפולציה לפי לגרנז' בנקודות אלו ונבצע אינטגרציה על הפולינום שיתקבל.
התוצאה שתתקבל:

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} y_i \cdot c_i$$

כאשר y_i הוא ערך הפונקציה בשורש ה-i של פולינום לז'נדר,
c_i הוא המשקל של השורש ה-i של פולינום לז'נדר.

כאשר הקטע הוא [a,b] נשתמש בהעתקה הבאה:

$$t = \frac{b-a}{2}\cdot x + \frac{b+a}{2}$$

אפרת וליאת

המשך-

מספר נקודות	שורש	משקל (c_i)
1	0	2
2	$\pm$0.57735	1
3	0	$\frac{8}{9}$
	$\pm$0.774597	$\frac{5}{9}$
4	$\pm$0.339981	0.652145
	$\pm$0.861136	0.347855
5	0	0.568889
	$\pm$0.538469	0.478629
	$\pm$0.90618	0.236927
6	$\pm$0.238619	0.467914
	$\pm$0.661209	0.360762
	$\pm$0.932469	0.171324

<u>משוואות דיפרנציאליות רגילות:</u>

<u>משוואות דיפרנציאליות מסדר ראשון:</u>

כאשר נתונה לנו משוואה בצורה $y'=f(y,t)$ ותנאי התחלה $y(t_0)=y_0$ נרצה למצוא את y, כאשר y היא פונקציה של t.

<u>שיטת אוילר:</u>

$y_{i+1}=y_i+h\cdot f(y_i,t_i)$ כאשר $y_{i+1}=y(t_{i+1})$ ו- $t_{i+1}=t_i+h=t_0+i\cdot h$.
ככל שנקטין את h נקבל דיוק גבוה יותר.
לאחר שחישבנו מספר הרצוי של הנקודות, נוכל לבצע אינטרפולציה לקבלת קירוב ל-y.

<u>משוואת אוילר הסתומה:</u>

במקרים מסוימים כמו במקרה של משוואה הומוגנית ($y'=cy$, c קבוע), שיטת אוילר תספק פתרון שגוי, ולכן נשתמש במשוואת אוילר הסתומה:

$$y_{i+1}=y_i+f(y_{i+1},t_{i+1})$$

על מנת להגיע לפתרון, יש להשתמש בשיטות נומריות למציאת שורשים, לדוגמא שיטת נקודת שבת.

בשיטה זו נאלץ לחשב עבור כל שלב מספר כלשהו של איטרציות, אך שיטה זו יותר יציבה משיטת אוילר הרגילה.

שיטת Runge kutta:

פיתוח לפי midpoint:

$$y_{i+1} = y_i + h \cdot f\left(y_i + \frac{h f(y_i, t_i)}{2},\ t_i + \frac{h}{2}\right)$$

כאשר $k_1 = h f(y_i, t_i) \qquad k_2 = h \cdot f\left(y_i + \frac{k_1}{2},\ t_i + \frac{h}{2}\right)$

תתקבל המשוואה הבאה:

$$y_{i+1} = y_i + h \cdot f\left(y_i + \frac{k_1}{2},\ t_i + \frac{h}{2}\right) = y_i + k_2$$

פיתוח לפי שיטת הטרפז:

$$y_{i+1} = y_i + \frac{h}{2} \cdot f(y_i, t_i) + \frac{h}{2} \cdot f(y_i + h \cdot f(y_i, t_i), t_i + h)$$

כאשר: $k_1 = h f(y_i, t_i) \qquad k_2 = h \cdot f(y_i + k_1,\ t_i + h)$

תתקבל המשוואה הבאה:

$$y_{i+1} = y_i + \frac{k_1}{2} + \frac{k_2}{2}$$

פקודה ב-matlab:

[T,Y] = ode45(odefun,tspan,y_0)

odefun- משוואה דיפרנציאלית y'=f(t,y).
tspan- וקטור המציין את גבולות האינטגרציה. אם קיימים יותר משני ערכים, יחזיר פתרונות בכל הערכים המבוקשים. עבור שני ערכים יחזיר את הפתרון המוערך בכל צעד אינטרציה.
y_0- וקטור תנאים התחלתיים.

T – וקטור של נקודות זמן.
Y – מערך פתרונות, כל שורה במערך מתאימה לפתרון בזמן שהוחזר בשורה המתאימה ב-T.